Enabling Secure Secret Updating for Unidirectional Key Distribution in RFID-Enabled Supply Chains

Shaoying Cai[1], Tieyan Li[2], Changshe Ma[1], Yingjiu Li[1], and Robert H. Deng[1]

[1] School of Information Systems, Singapore Management University
shaoyingcai.2009@phdis.smu.edu.sg,
{changshema,yjli,robertdeng}@smu.edu.sg
[2] Institute for Infocomm Research, A*STAR Singapore
litieyan@i2r.a-star.edu.sg

Abstract. In USENIX Security 08, Juels, Pappu and Parno proposed a secret sharing based mechanism to alleviate the key distribution problem in RFID-enabled supply chains. Compared to existing pseudonym based RFID protocols, the secret sharing based solution is more suitable for RFID-enabled supply chains since it does not require a database of keys be distributed among supply chain parties for secure ownership transfer of RFID tags. However, this mechanism cannot resist tag tracking and tag counterfeiting attacks in supply chain systems. It is also not convenient for downstream supply chain parties to adjust the size of RFID tag collections in recovering tag keys. To address these problems, we propose a flexible and secure secret update protocol which enables each supply chain party to update tag keys in a secure and efficient manner. Our proposal enhances the previous secret sharing based mechanism in that it not only solves the flexibility problem in unidirectional key distribution, but also ensures the security for ownership transfer of tags in RFID-enabled supply chains.

1 Introduction

Radio-frequency identification (RFID) is a wireless Automatic Identification and Data Capture (AIDC) technology that has been widely deployed in many applications including supply chain management. While RFID technology facilitates efficient management of RFID tags, it also triggers privacy concerns since sensitive information about RFID tags may be collected by an adversary via wireless communication channel without their owner's awareness. To prevent RFID tags from unwanted readouts, various solutions have been proposed in the literature. One solution is to send a *Kill* command [2] to a tag so that the tag is "dead" to any queries. Another solution is to physically clip a tag's antenna so that the tag is silenced [7]. Juels, Rivest and Szydlo introduced the blocker tag [6], which generates a signal that collides with all tags to be protected. Once the blocker tag is removed or disabled, however, the privacy of the protected tags may be disclosed. To make RFID tags always readily responsible, Fishkin, Roy

S. Qing, C.J. Mitchell, and G. Wang (Eds.): ICICS 2009, LNCS 5927, pp. 150–164, 2009.

and Jiang [3] proposed a distance-based access control scheme in which the tags reply with different levels of details depending on their distance to the reader, assuming that an adversary cannot get close enough to a tag as an authorized reader. However, it is difficult to prevent an adversary from getting close to a tag in many applications; consequently, the adversary may obtain sensitive information about some tags.

To address this concern, many protocols based on pseudonymous ID have been proposed [5], including the hash-lock of Weis et al. [15], the hash chain of Ohkubo, Suzuki and Kinoshita [12], the tree-based proposal of Molnar and Wagner [11], and many randomized pseudonym-based protocols such as the one proposed by Li and Ding [10]. In such protocols, each tag is associated with a pseudonymous ID. An authorized reader can identify the tag from its pseudonyms because it has access to a database that maps each tag's pseudonym to its real ID based on a secret key. Without the database, an adversary cannot identify a tag nor track it from its pseudonyms (which may change from time to time). When the tagged items change their ownership in a supply chain, the database needs to be transferred so that the new owner can recognize the tags according to the database. How to efficiently distribute the database among authorized parties is critical for applying pseudonym based protocols in RFID-enabled supply chains. Since the database consists of the keys for all tags, this problem is essentially a key distribution problem to be addressed in RFID-enabled supply chains.

In supply chain practice, especially for dynamic or ad hoc supply chain structure, the parties in supply chain are usually lack of secure network connections. A practical solution to the key distribution problem is to split a tag key into a number of shares and store the shares to multiple tags. Since the tag keys are stored in the tags directly, an authorized reader/party can collect enough shares and recover the keys, while an adversary is assumed to have limited access to the tags such that he/she cannot collect enough shares for recovering the keys. In this solution, there is no need of distributing a key database among supply chain parties. This secret sharing based solution is thus particularly useful for protecting dynamic and ad hoc supply chains.

A recent work in this direction is conducted by Juels, Pappu and Parno [4], which we call Juels-Pappu-Parno key sharing mechanism, or JPP mechanism for short. In this solution, a common key k of a batch of tags is split with a (τ, n)-secret sharing scheme, and each tag T_i stores a share S_i of k and its individual (encrypted) information M_i. A reader can recover k with access to at least τ shares. From k and M_i, a reader can decrypt the information about tag T_i. The reader is referred to Section 2.3 for more details about the JPP mechanism. This proposal does not restrict on the time period in which a tag should be read, or the number of the tags which should be attached to an item; thus, it is considered to be the only secret sharing based solution that is suitable for RFID-enabled supply chains.

However, the JPP mechanism is not secure in that a tag always sends a constant reply to any query. As a result, the tag is vulnerable to tracking by

either supply chain parties or outsiders. Also, an adversary can impersonate a tag in replay attack. When the tagged items are handed over from upstream parties to downstream parties in a supply chain, the downstream parties need to adjust the threshold in key recovery as they reassemble the collection/batch of tagged items. The JPP mechanism does not provide such flexibility.

In this paper, we propose a secret update protocol to address the problems of the JPP mechanism. Our secret update protocol enhances the security level of the JPP mechanism against tracking and impersonation attacks. It also makes it convenient for a supply chain party to adjust the threshold in key recovery according to the size of each batch of tags to be processed. Our work enhances the security and practicality of the JPP mechanism with reasonable cost paid to increase the tag's functionality.

The rest of this paper is organized as follows. In Section 2, we review the characteristics of supply chain, the secret sharing approaches, and the JPP mechanism. In Section 3, we elaborate on our secret update protocol. In Section 4, we formally prove the security of our secret update protocol. At last, we conclude this paper.

2 Reviews

In this section, we briefly introduce the background knowledge about the security requirements for RFID-enabled supply chains. We review several secret sharing approaches and revisit the JPP mechanism.

2.1 Security Requirements for RFID-Enabled Supply Chains

As pointed out in [4], the requirements for any security mechanism in RFID-enabled supply chains should be carefully defined according to the supply chain characteristics, which are summarized below.

- ◇ **None pre-existing trust relationship:** The two adjacent parties in a supply chain may have no previous trust relationship. The current owner of the tagged items may not always know which party will take over part or all of the tagged items before it gets the order from the next party.
- ◇ **Unidirectional downsizing:** Items start off in large collections and progressively get whittled down into smaller aggregates as they make their way from upstream parties to downstream parties.
- ◇ **Compulsory processing orders:** A batch of tags that are processed together by a downstream party must be processed together by an upstream party.

As such, a schematic representing the de- & re-packing of a case containing multiple item tags within a supply chain party is illustrated in Fig. 1 as our running example: *On receiving a case of* 100 *items transported from an upstream party, the current owner disassembles the case into item-level tags. According to some business orders from downstream parties, the owner then repackages those*

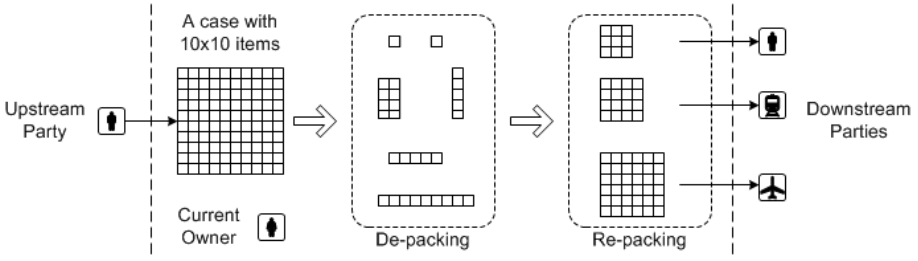

Fig. 1. De- & Re-packing of Item Tags in RFID-enabled Supply Chains. *A case of* 10×10 *items, transported from an upstream party, can be de-packed and re-packed by the current owner. The newly packed cases are to be shipped out to the downstream parties.*

items into smaller cases with variable sizes and ships them out. Before shipping out, the owner of the tags would most likely update the tags so that the smaller batches can be efficiently authenticated by the downstream parties.

2.2 Secret Sharing Approaches

Till now, three secret sharing mechanisms [8,9,4] were proposed to solve the key distribution problem in RFID-enabled supply chains.

The Shamir tag proposed by Langheinrich and Marti [8] is the first proposal in this direction. This solution splits the *true ID* of a tag using Shamir's secret sharing scheme [14] and stores all the shares on the tag itself. All the shares form the *new ID* of a tag. Upon a reader's inquiry, an initial set of random bits from the new ID is released, following by subsequent throttled single-bit releases. Eventually, all bits of the new ID will be released and only then can the true ID be computed [8]. The security of this proposal is relevant to the time that an adversary can access the tag. An RFID reader needs a sufficient period of time to collect all bits of a tag's new ID before it can recover the tag's ture ID. The time period is typically several minutes for reasonable security. The security of this solution depends on the assumption that a "hit-and-run" adversary cannot access a tag in a long enough time period to collect all bits of a tag's new ID. The problem is, in a supply chain, typically a large number of tags need to be processed in an efficient manner. It may not be practical to spend several minutes to identify a tag. Although the initial set of random bits can be used for fast identifying, it works only when the reader knows all tags' new IDs and true IDs. Since a database of tags is not distributed among supply chain parties (for solving the key distribution problem), a new owner of the tags has no knowledge about the tags' new IDs. Therefore, the Shamir tag scheme is not practical for ownership transfer in RFID-enabled supply chains.

Langheinrich and Marti have also extended Shamir's scheme to distribute an item's ID over hundreds of tags that are attached to or integrated into the item's material [9]. This method may be suitable for protecting containers or

large items; however, it is not practical to authenticate hundreds of tags for each item in supply chains.

In USENIX Security 08, Juels, Pappu and Parno proposed a key sharing mechanism (i.e., the JPP mechanism) [4] to enhance the practicality of the two early secret sharing based solutions proposed by Langheinrich and Marti [8,9] by removing the constraints on the period of each tag being read and the number of tags attached to each item. The JPP mechanism is particularly efficient for ownership transfer in RFID-enabled supply chains since it eliminates the need for distributing a database of tag keys among supply chain parties.

2.3 The JPP Mechanism

In the JPP-mechanism, a batch of tags share the same key k, which is split to n shares using a (τ, n) threshold secret sharing (TSS) scheme [14], where $\tau < n$ is a threshold. Anyone who collects at least τ shares can recover k. The JPP mechanism provides two methods to implement the (τ, n)-TSS scheme.

The first one is called "secret sharing across space" which uses Error Correcting Code (ECC) to construct the (τ, n)-TSS scheme. Each tag T_i stores a share s_i as well as its symmetrically encrypted information $E_k(m_i)$, where m_i is the information about the tag T_i. The tag T_i sends $(s_i, E_k(m_i))$ to any reader who queries it.

The other method is called "secret sharing across time", which uses the "Sliding-Window Information Secret Sharing" (SWISS) scheme to generate the shares of a tag. Each tag stores values of (s_{i1}, s_{i2}, r_i), where s_{i1} and s_{i2} are two key shares, and r_i is a tag-specific random number. One of key shares is used to derive the common key k of the tags in the timing window to which tag T_i belongs (the timing window determines which key share is used in deriving the key). From k and r_i, a reader can derive an individual key[1] k_i for each tag T_i, which is used to symmetrically encrypt the tag information m_i as $E_{k_i}(m_i)$.

We can see that the JPP mechanism is based on spreading a common secret into different tags. To summarize, we denote the tag's content as (S_i, M_i), where S_i represents the values used to derive the common key k, and M_i is the information related to the tag T_i itself. In the "secret sharing across space" method, we have $S_i = s_i$ and $M_i = E_k(m_i)$, while in the "secret sharing across time" method, we have $S_i = s_{i1} \| s_{i2}$ and $M_i = r_i \| E_{k_i}(m_i)$, where k_i is derived from the common key k.

The JPP mechanism is secure under the assumption that an adversary cannot get access to enough shares for recovering a tag key in the "open area" (e.g., retail stores or customer homes), while legitimate supply chain parties can collect enough shares for recovering each tag key in the "closed area" of a supply chain, to which the adversary does not have access.

[1] Note that although in the "secret sharing across time" method, each tag has a separate key k_i, the key k_i is derived from a common key k and r_i as $k_i = h(r_i, k)$, where h is a hash function. Since r_i is available to any reader, this method does not provide any stronger security than the "secret sharing across space" method in which all the tags shares a common key.

3 Secret Updating for Unidirectional Key Distribution

3.1 Observations

One of the highlights of the JPP mechanism is that it is suitable for EPCglobal Class-1 Generation-2 tags. The reason is that JPP mechanism requires no processing power on the tags and that the storage of EPCglobal Class-1 Generation-2 tags is enough to hold the tag information. This means that the JPP mechanism can easily be deployed in current RFID systems without changing the requirements to the tags. It also eliminates the need of distributing a database of tags among supply chain parties for efficient ownership transfer in RFID-enabled supply chains. However, we have two key observations over the JPP mechanism.

Firstly, the adversary model of the JPP mechanism is not too strong in that the tags are assumed to be secure within the "closed area" of a supply chain, while the tags are exposed to an adversary only in the "open area." In practice, the security of supply chain is arguable; the parties in a dynamic or ad hoc supply chain may not trust or even know each other before they transfer tags to each other. It is more reasonable to assume that the adversary's power is limited rather than the tags are secure. Under this assumption, we have the following security observation on the JPP mechanism.

Vulnerable to tracking: In the JPP mechanism, a tag T_i always sends the same reply (S_i, M_i) to any reader who queries it. Although an adversary may not get enough shares to decrypt the content of the tag, the never-changing reply can be used by the adversary to track the tag.

Vulnerable to counterfeiting: As the public accessible message (S_i, M_i) is used for a reader to identify the tag T_i, an adversary can easily fabricate a tag that also sends (S_i, M_i), and replace the tagged item with the fabricated tag.

Secondly, the realistic deployment of the JPP mechanism is largely restricted by the so called monopolistic key assignment model, in which a monopoly (typically the manufacturer of the goods) pre-assigns all the keys (shares) to the tags according a fixed secret sharing scheme with conjectured parameters. Under this assumption, the monopoly has to know or predict much detailed information on how goods are de-packed and re-packed by the downstream parties along a supply chain. If the threshold of the secret sharing scheme is set to be too large, it is possible that the batch size becomes smaller than the threshold so that a valid downstream party cannot promptly collect enough shares to recover the key k. If the threshold is set to be too small (so that all supply chain parties can easily collect enough shares), it may also be easier for an adversary to collect enough shares, especially in upstream supply chains. Without the knowledge that is either not known, or hard to predict, and subject to uninformed changes, a proper threshold is hard, if not impossible, to be decided. Thus, this *one-size-fits-all* solution is not flexibly applicable.

In the unidirectional supply chain channel, no one has better knowledge on how to de-pack and re-pack the tagged goods than the current owner. As in our running example shown in Fig. 1, *a* 10×10 *case, in which all the tags can be originally assigned with shares by a* $(33, 100)$*-TSS scheme, is de-packed and re-packed into smaller cases by the current owner such that the tags in a* 3×3 *case can be re-assigned with secret shares by a* $(3, 9)$*-TSS scheme, and the tags in a* 6×6 *case with shares by a* $(12, 36)$*-TSS scheme.* It is highly demanded that this current owner be able to update the tags according to the status quo. We thus promote a flexible unidirectional key distribution scheme in RFID-enabled supply chains, where an intermediate supply chain party can flexibly and securely update the secrets (or shares) on the tags that are currently in processing, with some secret update protocol as proposed below.

3.2 The Secret Update Protocol

In order to solve the security and flexibility problems of the JPP mechanism while maintaining its merits, we propose a flexible and secure secret update protocol as an enhancement of the JPP mechanism. The purpose is to change the tag message (S_i, M_i) by each supply chain party to prevent the tracking and counterfeiting across supply chain parties; the secret update protocol may also change the parameters of the secret sharing scheme to adapt to the changes in collection size in supply chains.

A tag must verify the validity of the reader when executing the secret update protocol, or else any malicious reader can rewrite the tag. In the JPP mechanism, the only difference between the valid reader and unauthorized reader is the valid reader can recover the key k. Consequently, a tag can verify the reader's validity by checking the reader's possession of k.

We realize the reader authentication by adding a value $c_i = h(k \| S_i)$ on the tag, where $h : \{0,1\}^* \to \{0,1\}^l$ is a one-way hash function, and l is the security parameter of the system [2]. In our scheme, c_i is used by the tag T_i to authenticate the reader. The secret update protocol updates all the values stored on the tag, namely (S_i, M_i, c_i), to a set of new values (S_i', M_i', c_i'). The requirement on tags is that the tags should have the ability to perform hash functions and have slightly more storage than the JPP mechanism to store c_i. This is the price to pay for the enhanced security and functionality.

Before the secret update protocol is launched by a reader, the reader is assumed to have recovered the key k with enough shares. It is also assumed that the reader has chosen a new key k' and split it using a pre-chosen secret sharing scheme with new parameters (τ', n'), where typically $\tau' \leq \tau$ and $n' \leq n$ for tags moving towards downstream supply chain. The secret update protocol is shown in Fig. 2 and described in the following.

1. **Reader $\longleftrightarrow$ Tag:** The reader first identifies the tag T_i based on the recovered key k and the shared secret c_i, with any privacy-enhanced

[2] The length of $(S_i' \| M_i')$ is supposed to be 96 bits in [4], then we set l to be 96 bits. Refer to Section 3.5 for more discussions on the implementation issues.

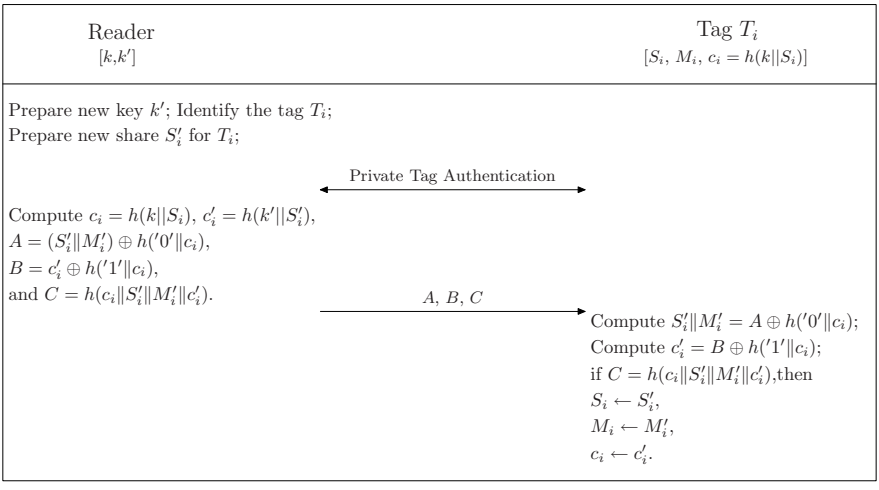

Fig. 2. The Secret Update Protocol

authentication protocol[3]. Once the tag is authenticated, the reader is ready to prepare the new secret share for the tag's updating.

2. **Reader $\longrightarrow$ Tag:** The reader then computes $c_i = h(k||S_i)$, and assigns an unused share S'_i (of the new key k') to T_i. The reader calculates $c'_i = h(k'||S'_i)$, $A = (S'_i||M'_i) \oplus h('0'||c_i)$, $B = c'_i \oplus h('1'||c_i)$, $C = h(c_i||S'_i||M'_i||c'_i)$. Finally, the reader sends (A, B, C) to T_i.
3. **Tag:** After receiving (A, B, C) from the reader, T_i computes $S'_i||M'_i = A \oplus h('0'||c_i)$, $c'_i = B \oplus h('1'||c_i)$. If $C = h(c_i||S'_i||M'_i||c'_i)$, the reader is authenticated. Then T_i updates its values to be $S_i = S'_i$, $M_i = M'_i$ and $c_i = c'_i$.

To make sure that a tag is updated successfully, the reader can identify the tag after the updating process. Recall that in the JPP mechanism, a tag T_i always sends the same reply (S_i, M_i) to the reader who queries it. The reader does not authenticate the tag; thus, an adversary can easily forge a tag by replaying the reply message. In comparison, our secret update protocol restricts this counterfeiting problem from propagating to different supply chain parties, it can be further strengthened to solve the problem even within a supply chain party's territory. To achieve this goal, the first step of the secret update protocol can be any privacy-enhanced tag authentication protocol. For instance, a typical challenge-response tag authentication protocol could be as follows: First, a reader sends its query together with a fresh random number r_1 to a tag. Second, the tag

[3] To propose any new private tag authentication protocol is not our major intention in this paper. As an example, a typical challenge-response protocol is described in the following paragraph. Note that if performance is the primary goal, a tag could be quickly identified without being privately authenticated by solely presenting its identifier.

T_i generates a fresh random number r_2 and replies (r_2, D) to the reader, where $D = h(r_1 \| r_2 \| c_i)$. The reader verifies the value of D based on its knowledge on c_i of T_i. Below we discuss the security, performance and implementation aspects of our protocol.

3.3 Security Properties

We list several primary security goals which are desired in a secret update protocol, and discuss how they are achieved in our protocol.

Authoritative access to tags. The security of the secret update protocol relies on the confidentiality of c_i. Given an update message (A, B, C), only the one who knows the value of c_i can obtain the values of S_i', M_i' and c_i'. Without knowing c_i, an adversary cannot forge a valid message (A', B', C') due to the security property of the hash function. Thus, unauthorized readers cannot update the tags' secret.

Authenticity of tags. In our updating protocol, the tag T_i is authenticated with any privacy-enhanced authentication scheme (*E.g.,* the challenge-response protocol, where D_i, the authentication message, can be verified only with the knowledge of c_i). The JPP mechanism does not provide tag authenticity since the tag always sends static reply to any query. Thus, our secret update protocol solves the counterfeiting problem.

Forward security. In our secret update protocol, the tag T_i is updated with new values (S_i', M_i', c_i'), which are totally independent from its previous values (S_i, M_i, c_i). Therefore, the protocol achieves forward secrecy for the tags' content in previous periods.

Untraceability. The JPP mechanism cannot resist tracking attacks, as the tag always sends static values S_i and M_i to any reader who queries it. Our secret update protocol provides the anti-tracking property by updating a tag's secret to a new value, thus an attacker is not able to link the values before and after the updating operation to the same tag (with the assumption that the updating messages are secure against eavesdropping, while an adversary is allowed to track a tag before and after the updating protocol). Note that even if an adversary eavesdrops the updating messages, as long as s/he cannot correlate the messages before and after the updating, s/he still cannot track a tag in different periods. However, if an active adversary is able to monitor the update process and query a tag immediately before and after its updating, s/he can track the tag by correlating S_i and S_i'. Note that such correlation is difficult to achieve in practice because the limited radio frequency range makes the protocol operate in a relatively secure environment, especially when update is performed.

Besides the enhancement of the security level, our secret update protocol solves the reassembly issue since the current owner of a tag can write new values (S_i', M_i') into the tag according to new secret sharing parameters (τ', n'). Thus, a downstream party may flexibly choose $\tau' < \tau$ and $n' < n$ for the convenience of

processing smaller batches of tags required in dynamic supply chain environment. The formal proof on the privacy of the tags can be found in Section 4.

3.4 Comparisons

Above we mentioned that secret sharing approaches present a new research direction on solving the key distribution problem in RFID-enabled supply chains. Their advantage can be demonstrated by comparing with several existing RFID authentication protocols that are selected to be classical and representative under a coarse classification. We list the major security and performance metrics of these schemes and compare our solution with them in Table 1.

Table 1. Security and Performance Comparisons

	Requirement for Key Distribution	Resistance to Tracking	Resistance to Counterfeiting	Complexity in Online Identification
Hash chain [12]	yes	yes	no	$O(\log n)$
Hash tree [11]	yes	yes	yes	$O(\log n)$
Pseudonym [10]	yes	yes	yes	$O(n)$
JPP mechanism [4]	no	no	no	$O(1)$
Our protocol	no	yes	yes	$O(1)$

From Table 1, the advantage of using secret sharing based mechanisms is very clear, as they simply solve the key distribution problem which is considered as a cornerstone of cryptography, and is also fraught with complexity in real world applications, while traditional schemes [12,11,10] presuppose the existence of shared keys between mutually trusted parties.

The hash chain-based scheme [12] is efficient since the online identification complexity is $O(\log n)$; however, it cannot resist counterfeiting attack. In the tree-based scheme [11], each tag stores $\log n$ secrets, in which $\log n - 1$ secrets are shared with other tags. It increases the storage requirement on the tags and communication rounds between the tag and the reader, although it also decreases the online identification time to $\log n$. The randomized pseudonym-based protocol [10] assigns each tag with a pseudonymous ID, which is mapped to a real ID stored in the backend database. The protocol can resist tracking and counterfeiting attacks, but its complexity of searching a tag is $O(n)$.

The JPP mechanism is the first applicable solution that is suitable for supply chains and without the need for pre-sharing of the secrets. However, it's security level is not sufficient as mentioned above. Our solution enhances the security of the JPP mechanism with additional computational requirements that are comparable with existing solutions, which typically require random number generator and hash function on the tag. In our protocol, the reader can obtain all tags' information after getting τ shares to recover the secret k. The reader also needs to conduct a private authentication protocol for identifying a tag. Besides, each tag needs to store three values which are constant on storage.

3.5 Implementation Considerations

In [4], Juels, Pappu, and Parno implemented a $(15, 20)$ threshold secret sharing scheme on Gen2 tags. In their case, for totally 20 available tags, a reader needs to collect at least 15 tags' shares to successfully recover the secret key and decrypt the encrypted information. The implementation only employs the 96-bit EPC memory bank of a "Alien Squiggle" Gen2 tag, of which 16 bits are used for storing a single share and 80 bits are used for storing the encrypted identity. Note that the shares are generated and written into the tags by encoding a secret key into 20 16-bit symbols using a $(20, 15)$ Reed-Solomon Error Correcting Code (RS-ECC) [13].

Given this parameterization (*w.r.t.*, $|S_i| = 16$ and $|M_i| = 80$), our protocol, replacing (S_i, M_i) with (S'_i, M'_i, c'_i), requires additional memory space for storing c'_i message (which is equivalent to 160 bits, if SHA-1 cryptographic hash function is used.). As such, we can not put them all into the EPC memory bank, but put (S'_i, M'_i) into the EPC memory bank as in [4], and put (c'_i) into the "User" memory bank [4]. Additionally, we comment that a typical EPC is exactly 96 bits, by applying a block cipher in any authenticated and encrypted method, the encrypted EPC message still has 96-bit length. Hence, a share value (S_i) in [4] might not be properly stored in the EPC memory bank (although its length is only 16-bit), but inevitably be stored in the user memory bank. Fortunately, unlike the memory banks for storing the *Tag unique IDentifier* (or "TID") and the *Access* and *Kill* passwords, both the EPC memory and the user memory have similar physical and deployment characteristics (regarding the *password-based lock, unlock, permalock*, and *password-based write* operations on these memory banks) according to EPCglobal UHF C1 G2 standard [2]. While the JPP mechanism requires only *write-once* EPC memory, our protocol requires *rewritable* EPC memory and user memory on a Gen2 tag. Such a *(re)write* operation is typically allowed in a **secured** state on interrogating a Gen2 tag, which is transitioned from an **open** state by providing the correct *Access* password. On implementing our protocol, we indicate that the 32-bit *Access* password of a Gen2 tag be individually derived from the recovered shared key k (e.g., we obtain the 32-bit *Access* password by hashing the key k concatenated with the tag's EPC code, and then taking the first 32 bits of the output.).

Moreover, in real-world usage, one has to determine the number of tags n processed in a batch and the threshold τ to recover the secret key. Now suppose the current owner in Fig. 1 receives a case of 100 tags, which are formatted with $(\tau, 100)$-TSS scheme. On choosing a proper threshold, τ can be set as the biggest value (e.g., $\tau = 80$) that is less than certain upper bound to maximally tolerate (up to 20) reading or erasure errors; alternatively, τ can be set as the smallest value (e.g., $\tau = 20$) that is greater than certain lower bound to guarantee the robustness on recovering the key with a minimum number (20) of tags. As such, the owner can recover the secret key and then decrypt the encrypted information

[4] Note that different tag manufactures will produce different form-factor Gen2 tags all conforming to EPCglobal C1 G2 specification [2]). E.g., a Philips UCODE EPC Gen2 tag contains 224 bits in the user memory bank.

attached with each tag. According to downstream parties' requirements, the owner can de-pack the case and re-pack it into several smaller cases including 4 cases with 4×4 tags and 1 cases with 6×6 tags. Similarly, a flexible $(4 \sim 12, 16)$-TSS scheme and a flexible $(8 \sim 28, 36)$-TSS scheme can be chosen for these two kinds of cases respectively, to either maintain robust recovery with minimal 20% available tags, or tolerate about 20% of reading/erasure errors.

4 Security Proof

In this section, we formally prove the security of our secret update protocol. At first, we formalize the security requirement for the secret update protocol. Then, we show that the proposed protocol satisfies the privacy requirement.

4.1 Security Model

We focus on the privacy property of our secret update protocol. As for the JPP authentication protocol running between an RFID reader and a batch of tags, it has been proven that in [4], it is infeasible for any polynomial time adversary to learn the shared key of the tags if the adversary cannot get access to the replies from at least τ tags. This privacy property is strengthened in our secret update protocol since the shared key of the tags can be updated by each valid supply chain party.

Informally, in the secret update protocol, there are two basic security requirements: (i) The former owner should not be able to trace the current owner. (ii) Any adversary outside the supply chain should not be able to link any two protocol messages in different *periods* (a period is the lifetime of a tag between its being updated by two adjacent owners). A formal privacy definition is described in the following privacy game of secret update protocol (Game PoT for short, which is illustrated in Fig. 3.).

We first give a formal description of an RFID system. In an RFID system, there are a set of tags $\mathcal{T} = \{T_1, \cdots, T_\ell\}$, a set of readers $\mathcal{R} = \{R_1, \cdots, R_m\}$ and an adversary $\mathcal{A}$. Each tag stores a secret which is updated when its owner has changed. The RFID system is initialized and updated through a (τ, n) secret sharing scheme [14]. To identify a tag, a reader interacts with the tag through the authentication protocol $\pi(R_i, T_j)$. When a tag is passed from one owner to another owner, its current owner R_i runs the secret update protocol $\kappa(R_i, T_j)$ to reset the internal state of tag T_j.

As above, the Game PoT consists of three phases. In the setup phase, the game initializes the RFID system. Then in the learning phase, the adversary $\mathcal{A}$ performs a series of queries to enlarge its knowledge base about the RFID system. In the third phase, the adversary $\mathcal{A}$ chooses two tags for challenging. Then, a tag is chosen by randomly updating one of the two tags. After this, the updated tag is given to the adversary as a challenging tag for him to distinguish it from the original two tags.

In the Game PoT, the adversary $\mathcal{A}$ is allowed to eavesdrop the protocol messages (by invoking an authentication protocol $\pi(R_i, T_j)$ between a reader R_i and

Game PoT

⟨**Game PoT**⟩

Setup:

(1) Initialize a set of tags $\mathcal{T} = \{T_1, \cdots, T_\ell\}$.
(2) Initialize a set of readers $\mathcal{R} = \{R_1, \cdots, R_m\}$.

Learning phase:

(3) For any $R_i \in \mathcal{R}$ and $T_j \in \mathcal{T}$, $\mathcal{A}$ may do the following in any order as long as $r_{k,pd} + v_{k,pd} \leqslant \tau$, where $r_{k,pd}$ and $v_{k,pd}$ denote the number of authentication protocol calls related to tags whose data was encrypted by k in the *period pd* and the number of corruption calls related to tags whose data was encrypted by k in the *period pd* respectively:
  (3.1) Make $\pi(R_i, T_j)$ calls, without exceeding r overall calls.
  (3.2) Make $\kappa(R_i, T_j)$ calls, without exceeding u overall calls.
  (3.3) Corrupt tags, without exceeding v overall calls.
  (3.4) Corrupt any reader except the reader R'.

Challenge phase:

(4) $\mathcal{A}$ selects two tags T_p and T_q.
(5) Let $T_0 = T_p$, $T_1 = T_q$ and $b \in_R \{0, 1\}$.
(6) the game runs $\kappa(R', T_b)$ and then sends T_b to $\mathcal{A}$.
(7) For any $R_i \in \mathcal{R}$ and $T_j \in \mathcal{T}$, $\mathcal{A}$ may do the following in any order as long as $r_{k,pd} + v_{k,pd} \leqslant \tau$:
  (7.1) Make $\pi(R_i, T_j)$ calls, without exceeding r overall calls.
  (7.2) Make $\kappa(R_i, T_j)$ calls, without exceeding u overall calls.
  (7.3) Corrupt tags, without exceeding v overall calls.
  (7.4) Corrupt any reader except the reader R'.
(8) $\mathcal{A}$ outputs a guess bit b'.
(9) If $b' = b$ then output 1, else 0.

Fig. 3. Privacy Game of Secret Update Protocol

a tag T_j) and to corrupt the tags under the restriction that the sum of the number of calls of the authentication protocol and the number of tag corruptions is at most τ (i.e., the threshold of the secret sharing scheme) during the same *period*. The adversary $\mathcal{A}$ is also allowed to invoke secret update protocol $\kappa(R_i, T_j)$ with restriction that it cannot learn the messages in the secret update protocol. Note that this assumption is commonly used in most secret update protocols and it is reasonable since the secret update protocol is usually executed in a relative secure environment such as the warehouse of a supply chain party. The adversary $\mathcal{A}$ is allowed to corrupt any readers (and hence to get their internal states) except the reader R' to which the challenging tag is presented.

The goal of the adversary $\mathcal{A}$ in the Game PoT is to distinguish between two different tags chosen by itself. The adversary may know the internal state of both tags. But the challenging tag is chosen by updating a tag selected randomly from the two tags.

Definition 1. *A function $f : N \to R$ is said to be negligible if for every $c > 0$ there exits a number $m \in N$ such that $f(n) < \frac{1}{n^c}$ holds for all $n > m$. Also, a function $f : N \to R$ is said to be overwhelming if $1 - f(n)$ is negligible.*

Definition 2. *The advantage of adversary $\mathcal{A}$ in the Game PoT is defined as*

$$\mathbf{Adv}_{\mathcal{A}}(r, u, v, \tau, n, \ell, m) = |2Pr[\textit{Game PoT outputs } 1] - 1|.$$

Definition 3. *We call that the RFID system is* $(r, u, v, \tau, n, \ell, m, t, \epsilon)$*-private, if there exists no polynomial probabilistic time adversary* $\mathcal{A}$ *whose advantage* $\mathbf{Adv}_{\mathcal{A}}(r, u, v, \tau, n, \ell, m)$ *is at least* ϵ *and whose running time is at most* t *in the Game PoT.*

4.2 Privacy of the Secret Update Protocol

The privacy of our secret update protocol relies on the privacy of the underlying secret sharing scheme. We now prove that our secret update protocol is $(r, u, v, \tau, n, t, \epsilon)$-private if the underlying secret sharing scheme is $(\tau, n, t, \epsilon/2)$-private. The following theorem characterizes the security of our secret update protocol[5].

Theorem 1. *In the random oracle model, if the secret sharing scheme used in our RFID protocol is* $(\tau, n, t, \epsilon/2)$*-private, then the proposed RFID system is* $(r, u, v, \tau, n, \ell, m, t, \epsilon)$*-private.*

5 Conclusion

In this paper, we design a flexible and secure secret update protocol as an extension to Juels, Pappu and Parno's unidirectional key distribution scheme [4]. Our secret update protocol provides desirable flexibility so that downstream supply chain parties can adjust the threshold in key recovery for the convenience of processing smaller batches of tags. Moreover, the secret update protocol makes it particularly difficult for the tags to be tracked across multiple supply chain parties. Compared with previous works, our solution is similar in that it does not require a database storing keys of tags being distributed to different supply chain parties, while it is more secure against tracking and counterfeiting attacks, and is more flexible in addressing the reassembly problem. Although our secret update protocol requires more powerful tags than the EPCglobal Class-1 Gen-2 tags, it is worth to pay for the enhanced security and functionality. Our future work will focus on the minimization of the cost of tags in our design.

Acknowledgment. This work is partly supported by A*Star SERC Grant No. 082 101 0022 in Singapore.

References

1. Cai, S., Li, T., Ma, C., Li, Y., Deng, R.: Enabling secure secret updating for unidirectional key distribution in rfid-enabled supply chains, `http://icsd.i2r.a-star.edu.sg/staff/tieyan/SecureRFID/docs/ICICS09-full.pdf`
2. EPCglobal. EPC radio-frequency identity protocols class-1 generation-2 UHF RFID protocol for communications at 860 MHz-960 MHz, version 1.2.0. (October 2008)

[5] The privacy proofs of the secret sharing protocol and the secret sharing scheme are eliminated here due to page limit, but are provided in the full version [1].

3. Fishkin, K., Roy, S., Jiang, B.: Some Methods for Privacy in RFID Communication. In: Castelluccia, C., Hartenstein, H., Paar, C., Westhoff, D. (eds.) ESAS 2004. LNCS, vol. 3313, pp. 42–53. Springer, Heidelberg (2005)
4. Juels, A., Pappu, R., Parno, B.: Unidirectional key distribution across time and space with applications to rfid security. In: 17th USENIX Security Symposium, pp. 75–90 (2008)
5. Juels, A.: RFID Security and Privacy: A Research Survey. IEEE Journal on Selected Areas in Communications 24(2), 381–394 (2006)
6. Juels, A., Rivest, R., Szydlo, M.: The Blocker Tag: Selective Blocking of RFID Tags for Consumer Privacy. In: Conference on Computer and Communications Security – ACM CCS 2003, October 2003, pp. 103–111 (2003)
7. Karjoth, G., Moskowitz, P.: Disabling RFID Tags with Visible Confirmation: Clipped Tags Are Silenced. In: Workshop on Privacy in the Electronic Society – WPES 2005 (November 2005)
8. Langheinrich, M., Marti, R.: Practical Minimalist Cryptography for RFID Privacy. IEEE Systems Journal, Special Issue on RFID Technology 1(2), 115–128 (2007)
9. Langheinrich, M., Marti, R.: Rfid privacy using spatially distributed shared secrets. In: Ichikawa, H., Cho, W.-D., Satoh, I., Youn, H.Y. (eds.) UCS 2007. LNCS, vol. 4836, pp. 1–16. Springer, Heidelberg (2007)
10. Li, Y., Ding, X.: Protecting RFID Communications in Supply Chains. In: Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security ASIACCS 2007, Singapore, pp. 234–241 (2007)
11. Molnar, D., Wagner, D.: Privacy and Security in Library RFID: Issues, Practices, and Architectures. In: Conference on Computer and Communications Security – ACM CCS 2004, October 2004, pp. 210–219 (2004)
12. Ohkubo, M., Suzuki, K., Kinoshita, S.: Efficient Hash-Chain Based RFID Privacy Protection Scheme. In: International Conference on Ubiquitous Computing – Ubicomp 2004 (September 2004)
13. Reed, I.S., Solomon, G.: Polynomial codes over certain finite fields. Journal SIAM 8, 300–304 (1960)
14. Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (1979)
15. Weis, S., Sarma, S., Rivest, R., Engels, D.: Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In: Hutter, D., Müller, G., Stephan, W., Ullmann, M. (eds.) Security in Pervasive Computing. LNCS, vol. 2802, pp. 454–469. Springer, Heidelberg (2004)